

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

TITLE OF THE INVENTION

JAVA PRINTER

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application claims priority to provisional U.S. patent application serial number 60/017,398, the contents of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

This invention relates to a printer or printer system using the Java language to control rasterizing an image and to control printing.

DESCRIPTION OF THE BACKGROUND

The Internet is undergoing explosive growth and many new technologies are being developed to keep up with this growth. Previously, in order to develop applications quickly, application developers sought to use specialized cross-platform application development techniques which create applications for multiple hardware and software platforms. For example, operating systems (i.e. Solaris, Windows 95, Windows 3.x, OS/2 and Unix) create applications and executable files differently and have been implemented on various processor types (Intel processors, 680x0, Power PCs, and Sun SPARCs). To develop applications for all permutations of operating systems and hardware is difficult and extends the product development cycle. To make cross-platform development faster and easier, Sun Microsystems developed a language

called Java which is object-oriented but simple. Java is described in Java in a Nutshell: A Desktop Quick Reference for Java Programmers by David Flannagan, published by O'Reilly & Associates, Inc., as well as in the Sun Series of books published by Prentice Hall Books entitled Core Java, Instant Java, Java by Example, and Just Java by Cornell, Pew, Jackson and Van Der Linden, respectively, which are incorporated herein by reference. One of Java's advantages is that it is a portable language which is independent of operating systems and hardware architectures. Further, applications developed using Java are adaptable or extendable using Java's ability to download new classes dynamically and to add the downloaded classes to an existing class hierarchy. Java also provides the advantages of distribution, language interpretation, security, high performance and a multi-threaded implementation.

Java enables applications to be written using an extensible set of objects, with each set of objects being defined in a separate group of objects called a package. The core set of objects for Java are defined in the java.lang package and describe the most central characteristics of the Java language. One of Java's advantages is that the character type that Java uses is the Unicode standard which allows English and Asian characters to be represented consistently and together in applications or documents generated using Java.

Other languages have been used to represent the layout of documents as they appear on printers. PostScript by Adobe is an extendable page-layout language which supports text and graphics on the same page. Some aspects of PostScript are described in PostScript by Example by Henry McGilton and Mary Campione, published by Addison-Wesley Publishing Company, the contents of which are incorporated herein by reference. PostScript uses stacks and dictionaries to extend the language. Some PostScript printers also have been equipped with non-volatile memories which are used to store configuration information for the printers. However, PostScript's lack of strong security features has enabled malicious users of the printer to update the parameters stored in the non-volatile memory, thereby disturbing the printer's use in network environments.

Other printer languages, such as PCL by Hewlett-Packard, have evolved from uni-directional, dot-matrix line printers and therefore lack many of the operators needed to control the placement of images on a page. PCL also lacks modularity. The macros defined by PCL use globally scoped variables that can affect the performance of other macros defined by the language.

Currently, all of the applications that print from host systems have to convert their internal document format to PostScript or PCL and then download the document to the printer using a printer driver designed to work with the specifics of the connected printer. Since there is a wide

variety of printers that can be used, each with a slightly different set of features and/or bugs, a large number of printer drivers have traditionally been shipped with applications, even though end-users actually only need the printer drivers for their specific printers. Furthermore, using conventional printing techniques, an inadvertent change in the printer driver used could cause the printer to print out the commands which describe how a page is to be laid out rather than interpreting the commands and rendering a resulting image. Further, for printers which support downloadable fonts, downloading of fonts often has been restricted to downloading to the printer's RAM, ROM font cartridge or attached hard disk.

This model of application and printer driver interaction has created an increase in work performed by end-users because of the inflexibility and limited communication capability of the printer when communicating with the application.

SUMMARY OF THE INVENTION

It is an object of the present invention to overcome at least one of the deficiencies described above in the implementation of a printer language.

It is a further object of the present invention to provide a printer which uses the Java language to interpret page-layout requests.

It is another object of the present invention to provide bi-directional communication between an application or printer

driver and a Java printer to enable a user to define how a page or series of pages should be laid out.

It is yet another object of the present invention to provide a World Wide Web interface to control a Java printer of the present application.

It is a further object of the present invention to use the Java-specific features of object-orientation, distribution, interpretation, security, architecture and neutrality, portability, performance, multi-threadedness, and dynamic loading to implement an improved printer.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete appreciation of the invention and many of the attendant advantages thereof will be readily obtained as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, wherein:

Figure 1 is a layout diagram of a Java printer system according to a first embodiment of the invention;

Figure 2 is a schematic illustration of a Java printer system according to the second embodiment of the present invention;

Figure 3 is a schematic illustration of a Web browser interface for configuring a Java printer of the present invention;

Figures 4 and 5 are schematic illustrations of a queue manager for controlling printing using a Java printer of the present invention;

Figure 6 is a schematic illustration of a Java applet in a Web browser that is used to control the pagination of a document sent to a Java printer;

Figure 7 is an HTML document used to identify to a Java-compatible World Wide Web browser that a queue manager applet should be loaded and run; and

Figure 8 is a schematic illustration of a class which extends the printer classes of the Java language to implement printing on letterhead.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring now to the drawings, wherein like reference numerals designate identical or corresponding parts throughout the several views, Figure 1 is a view showing a layout of the Java printer system of the present invention. A conventional laser printer 100 is connected to a print server 102 via a high speed communication link 106 (i.e. SCSI bus), and print server 102 receives jobs via an external communication link 104 which can be a network link (Ethernet, token ring, ATM), an ISDN connection, or a modem connection. The print server 102 can be any general purpose computer system capable of running a Java interpreter, and specifically includes at least a central processing unit (CPU), random access memory (RAM), a mass storage device (i.e., a hard disk, a magneto-optical disk), an input device (i.e., mouse, keyboard, touch screen) and an output device (i.e., monitor, heads-up display, virtual reality headset). Further, the print server 102 includes any of the available commercial operating systems (i.e., UNIX, Windows 95, Windows NT, OS/2, Linux) and any other software required to implement network or Internet communications.

In the first embodiment, Java print requests are received by the print server 102 via the external communication link 104 through a socket listening on a designated port (i.e., port 80 for hypertext transfer protocol), and the print server 102 converts the received print request from a Java request to a printer request for the attached laser printer 100 in the printer language of the attached laser printer 100. The print

server 102 also can receive standard hypertext transfer protocol (HTTP) requests and produce World Wide Web (WWW) pages as a result or update the configuration of the laser printer 100 if the HTTP request has been sent by a system administrator or an authorized user.

As shown in Figure 2, the laser printer 100, print server 102 and communication link 106 can be combined into a single network printer which is a Java printer 110 connected to the external communication link 104. In the second embodiment, the Java printer 110 receives print requests or WWW/HTTP requests directly by listening on the appropriate port. Since Java is architecture neutral, the creation of a Java printer 110 and its corresponding classes can quickly be ported or migrated to a new processor used in a new laser printer when the new processor becomes available. Further, when Java processors become available which execute Java Unicode instructions directly, the Java printer 110 can be implemented with a Java processor. Any Java printer 110 should conform to the Java Virtual Machine Specification. The August 21, 1995, Release 1.0 Beta Draft version of the JVM specification is incorporated herein by reference.

Java has a rich set of graphics operators that match PostScript and PCL, and since Java is an object-oriented language which provides extensions, new complex graphics operators can be created which are subclasses of the existing graphics primitives, thereby allowing complex images to be described compactly. The present invention extends the

Graphics class of the java.awt package to control drawing of images on pages, by implementing a new class, Printer. Each of the other methods of the Graphics class would likewise be implemented to allow colors and fonts to be changed and to allow lines and filled and empty polygons to be drawn. For example, calling java.awt.Printer.drawString(msg, x, y) would cause the string "msg" to be drawn on a page at position (x,y). The Printer class also would implement a function, similar to PostScript's "showpage" command, which signals that a complete page has been rendered and that the resulting image should be transferred to Java laser printer 110. Further, based on the built-in security, network capability and multi-lingual support of Java, a Java printer can handle inputs from different platforms in multiple languages and create the desired documents.

As shown in Figure 3, a system administrator who wishes to configure a Java printer 110 can use a standard WWW browser (i.e., Netscape Navigator, Mosaic, Microsoft Navigator, IBM Web Explorer) to remotely configure the Java printer 110. The Web browser screen 120 includes a title bar 122, a menu bar 124, button icons 126, a document identifier text entry box 128 and plural additional controls such as drop box 130 which includes a list of Internet connections for which a configuration can be established. A system administrator would authenticate himself/herself to the Java printer 110 using any available security technique (i.e., Secure Socket Layer, Public Key Encryption, Symmetric Key Encryption, or a

User ID and Password Hash), wherein the public key, private key or valid user ID and password hashes are stored in the non-volatile memory of the Java printer 110 or in remote local with which the Java printer 110 can communicate securely. Having authenticated himself/herself, the system administrator would choose the configure document of printer1 as the document to be opened by specifying the appropriate URL in the text entry box 128, such as specifying:

http://printer1.companyname.com/configure.

When the Java printer 110 which was listening on the appropriate port receives the request for this document, the Java printer 110 would parse the name of the document requested from the rest of the request and determine that the system administrator wished to configure the printer for a particular user. (The complete specification for the format of an HTTP request or response can be found in the HTTP standard (versions 1.1 or 1.0). Information on HTML can be found in Using HTML: The Definitive Guide by Musciano and Kennedy, and information on Internet information services can be found in Managing Internet Information Services by Liu et al. The contents of these references are incorporated herein by reference.) In response to a request for a document, the Java printer 110 would send back to the browser, via the socket used to send the request, a response like the page shown in Figure 3 and referenced by reference numeral 131. The Java printer 110 also may use other information in the request to determine if the default response should be

modified. The response could be modified to include text in the system administrator's native language, or similar language/cultural changes. The system administrator could then choose which user's Internet number the configuration is for by selecting an Internet number from combination box 130. As would be evident to one of ordinary skill in the art, this combination box could be replaced by a select box such that multiple configurations for multiple Internet addresses could be set simultaneously. The returned page 131 allows the system administrator to establish paper size by choosing one of the radio buttons selecting A4 paper, 8.5 x 11 paper or legal size paper. Also, the default tray is specified, allowing the system administrator to choose between letterhead paper and plain paper.

Additional values can be set, as shown in the Figure 3 and as well as those parameters that would be evident to one of ordinary skill in the art in light of the present invention. In fact, any parameter usually set by buttons on a printer can be set through the standard WWW interface. In addition, on-line information which can not be easily displayed using LCD screens on the printer can be provided by selecting a hypertext link 134. This help information can either point to local help or remotely stored help, such as might be stored at the WWW site of the manufacturer of the Java printer 110.

Further, the Java printer 110, or the combination print server 102 and laser printer 100, can utilize the external

communication link 104 dynamically to retrieve documents or portions of documents from other web sites for printing or for otherwise modifying the operation of the Java printer 110, such as loading updates to printer code. The Java printer 110 can likewise use the external communication link 104 for downloading fonts from remote locations whenever the font is needed. To download fonts, changes, or other information, the Java printer 110 can use any transfer protocol implemented by an included Java package, including HTTP, FTP, Gopher, etc. By specifying the URL from which the information should be obtained and using the `java.net.getContent()` method information can be received as a String. In addition, for new or previously unimplemented protocols, a `java.net.URLConnection` object would be used to receive and parse the contents of a new document type. As described earlier, this provides an advantage over PostScript printers which have to have fonts loaded in their RAM, ROM or hard disk in order to use them. Table 1 below shows some of the advantages of using Java over HP/PCL and PostScript.

Table I

Network Configuration	Partially	Partially	Yes
Network Security	No	No	Yes
Code Size	Small	Big	Small
Ability to Handle Double-Byte Characters	Difficult	Difficult	Easy

Customization	Difficult	Difficult	Easy
Resources	One Location	One Location	Anywhere on the Network
Printer Driver	Printer Specific	Printer Specific	Universal
Graphics Operation	Difficult	Medium	Easy
Extensibility	Difficult	Difficult	Easy
Upgrade	Difficult	Difficult	Easy
Remote Diagnostics and Maintenance	Very Difficult	Very Difficult	Easy

As shown in Figure 4, the Java printer 110 can additionally be configured with a queue manager which can further control the Java printer 110 using a standard WWW interface 120. Figure 4 shows that a system administrator has requested queue information from Java printer 110 and received back an applet which provides configuration information to the system administrator. When the system administrator uses URL:

<http://printer1.companyname.com/queue>,

the web browser 120 changes the title bar 122 to reflect that the requested page represents the queue manager. The returned page 136 includes a series of graphics 160 which represent documents, their titles, their print times and the Internet addresses from which the jobs were submitted. These graphics 160 are displayed in a scrollable window 144 controlled by a scroll bar 138. Further, the applet includes a filter with an associated combination box 140 for filtering the graphics 160 which are displayed in the scrollable window 144. As shown in Figure 5, when an Internet address is used as a filter, all

documents not submitted by a specified internet address (i.e., 123.45.67.89) are removed from the scrollable window 144. Referring back to Figure 4, the applet also provides buttons 146, 148, 150, 152, and 156 which control how the Java printer 110 processes the documents. To control a print job, any of the graphics 160 can be highlighted and then one of the buttons 146, 148, 150, 152 and 156 depressed. The queue manager then will modify the queue characteristics for the highlighted job(s) according to the button which was depressed.

Since Java is multi-threaded, the Java printer 110 can multi-task between any of its functions (i.e., printing the current job, receiving a new job, pausing a job, killing a job, reordering the job such that a job is made the next job to be printed or the last job to be printed, displaying characteristics of a job, resuming a paused job, setting default configuration information or determining the status of the printer). Conventional printers perform one job at a time and do not have such a printing capability.

As shown in Figures 4 and 5, another advantage of a Java printer 110 is that status information 139 can be requested while other activities are being performed. Although the status information 139 is only a snapshot of the current status, the applet can periodically update the status information 139 using the network capabilities of Java and the browser provided by the DatagramPacket, DatagramSocket and Socket classes, as implemented by the java.net package.

Furthermore, since the Java printer 110 performs active multitasking, the Java printer 110 can also periodically send status reports via to system administrators or other users by any other implemented protocol (i.e., SMTP, FTP).

As shown in Figure 6, the Java printer 110 is also capable of performing interactive pagination based on an application using Java as the print language. The Java printer 110 can either be used as a traditional printer whereby the application specifies the pagination and sends print requests to the Java printer 110 in preformatted pages, or the application wishing to print uses the enhanced capabilities of the Java printer 110 to request that the Java printer 110 create a default pagination based on the characteristics of the printer.

Figure 6 shows a document which has been paginated by a Java printer 110. Having sent Java code in the form of a WWW/HTTP request to the Java printer 110, the pagination applet, or a portion of the application written to support Java, displays the result as specified by the Java printer 110. In Figure 6, the document has been split into four thumbnail sketches 170 which depict the layout of the document as established by the Java printer 110. The four thumbnail sketches 170 show that a figure is split across the second and third thumbnail sketches 170 into two parts 172a and 172b. Further, a paragraph of text has been split between the first and second pages, leaving a single line 171a separate from the rest of the text 171b. To modify the default pagination, the

application places page arrows 173 next to the thumbnail sketches 170 to indicate where the user wishes the pagination to actually occur. Using page arrows 173, the single line of text of 171a can be grouped with the rest of the text 171b, and the image 172a and 172b can be grouped. When the user has set the page arrows 173, the user uses the Resubmit button 174 to send the new configuration Java printer 110 which recreates thumbnail sketches 170 and returns the new result to web browser or pagination output/application. If the thumbnail sketches 170 were satisfiable to the user, the Ok button 176 would be used to signal that the pagination has been accepted by the user and that the pages should be printed. In addition, the Cancel button 178 can be depressed if the user wishes to cancel the printing. This allows the Java printer 110 to clear the Java print request from the pending queue of outstanding paginations. By explicitly clearing this information, the garbage collection process of the Java interpreter can reclaim the memory of the Java printer 110 more quickly. Although Figure 6 has been described in terms of page arrows 173, any type of pagination identifier could be used to describe how pages should be repaginated. For example, a continuous, scrollable image could be presented, and the user would use the applet/application to draw lines where the pagination should be. Furthermore, the standard hard return tag of HTML, <HR>, can be used in an alternate embodiment to show where page breaks occur. In addition, the thumbnail sketches 170 can be any type of graphic that can be

displayed on a page 120. These graphics include tnumonall GIFs, Java images, etc., several of which are supported by the java.image package.

As shown in Figure 7, the applet of Figures 4 and 5 can be generated using standard HTML code with a browser which supports Java. Using the APPLET keyword, the browser knows to load the queue manager class and display the window with a width of 500 and a height of 300.

As shown in Figure 8, the Java printer 110 can be extended with new classes such as the Letterhead class of Figure 8. The Letterhead class extends the printing class which provides the base printing functionality of the present invention. Although the Letterhead class shown is not completely illustrated, it would be evident to one of ordinary skill in the art that the Letterhead class would be used to first print the letterhead text onto a page to be printed, and then the Print class would perform its printing of the rest of the page. The illustrative class allows the printing on two types of letterhead, cover sheets and everything else, by specifying different strings and positions for the two pages. The Letterhead class could be extended in an alternate embodiment to use arrays, as supported by the Java language, to allow multiple strings and positions to be stored for each letterhead page.

Furthermore, to smooth the transition from PostScript and PCL to Java, Java printers can be implemented which utilize subclasses of the Printer class that receive and render

PostScript, PCL or both. These classes can even be added dynamically by the system administrator when they become available. The new classes would be loaded using a URL specified by the system administrator. On the otherhand, traditional printer drivers which utilize GDI calls can be modified to generate Java code from the received GDI application. For example, since windows documents are generated using standard GDI calls, a Java printer 110 can be selected like any other supported printer, and the Java printer driver then would convert the Windows GDI calls into Java code to be sent to the Java printer 110.

Obviously, numerous modifications and variations of the present invention are possible in light of the above teachings without departing from the intended scope of the present invention. Such changes include storing the user profile information in remote databases rather than in the Java printer 110 itself. By using the Java SQL API, also known as the JDBC, remote databases can store the user profile information, and applets still can query and update the user profile information. Since the JDBC specification has not been finalized, the version 0.70 draft specification dated May 7, 1996 is incorporated herein by reference.